"Express Mail" Mailing Label No. EV 327132365 US

February 19, 2004
Date of Deposit

Patent
Our Case No. 7871/17

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTORS: DAVID J. COLLODI

TITLE: PHYSICS ENGINE

ATTORNEY: WILLIAM F. PRENDERGAST
(Reg. No. 34,699)
BRINKS HOFER GILSON & LIONE
POST OFFICE BOX 10395
CHICAGO, ILLINOIS 60610
(312) 321-4200

PHYSICS ENGINE

FIELD OF THE INVENTION

[0001] The present invention relates to the calculation of the physical dynamics of a video/computer game. More particularly, the present invention relates to an improved physics engine that provides a more efficient process with heightened realism.

BACKGROUND OF THE INVENTION

[0002] The design of a real-time video/computer game is full of challenges. One of them is developing an efficient and realistic approach to calculation of the physical dynamics of the game. Real time games demand the most from the processor of a computer. The calculations required to conduct the game demand the most from even the faster game processors available. For example, the calculation of physical dynamics of a game environment can result in lengthy computations. This is particularly true where the game involves multiple interacting objects. Where compromises are required, many conventional games tend to provide simplified approaches to the calculation of the physical dynamics or physics of the game. However, these compromises often lead to a game that can lack realism.

[0003] Enhancing the realism, and particularly the physical dynamics of a game is often not straightforward. The implementation of a sophisticated and robust physics engine presents a number of technical challenges which have not been entirely overcome by prior art approaches. Some approaches use linear and quadratic programming to minimize sets of functions, while other approaches employ pivoting techniques from linear complementary problems to obtain solutions. These techniques, however, suffer from a high order solution complexity and runtime and are, therefore, only applicable to video games in a limited sense. Other approaches have involved the use of simplified physics engines which require fewer computations, but come at the expense of a more realistic game experience. Even inexperienced game players may find that a

simplified approach to physical dynamics results in a relatively artificial game experience.

[0004] Therefore, there is a need for a physics engine capable of providing a computationally efficient process while not sacrificing the realism of the game environment.

SUMMARY OF THE INVENTION

[0005] The present invention is directed to a method of simulating the physical dynamics of a predetermined set of objects that are part of a computer/video game. The objects are connected to each other at one or more respective links where some of the links represent hard contact between separate objects. The steps of the invention include creating a nested grouping of a plurality of binary objects. The objects are defined by a plurality of polygons. Starting with the most deeply nested binary object and proceeding outward, a solution is solved for the physical dynamics of the objects in the binary objects at the respective links.

[0006] A further aspect of the invention involves solving a constrained solution for the physical dynamics of a set of objects connected to each other at one or more respective links. One or more link weight values are assigned to each link such that the link weight values constrain the solution. A first iterative solution is then solved for the dynamics of the objects using the weight values. The link weight values are then adjusted for links which violate the solution constraints. Multiple solution iterations are then performed with one or more link weight values adjusted at each iteration until the solution conforms to the constraints within a predetermined acceptable tolerance.

[0007] An additional aspect of the present invention is directed to a video game system with improved physical dynamics comprising a binary division unit operable to create a nested grouping of binary objects from a set of objects connected by one or more respective links. The video games system further comprises a dynamics unit with one or more processors operable to solve a solution for the physical dynamics of the objects.

[0008] It should be recognized that the steps of the present invention may be practiced in an order other than that identified in the claims. Accordingly, the

order of the steps in the method claims provided herein should not be construed as in any way limiting the steps of the present invention to a particular order.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Figure 1 is a diagram illustrating an embodiment of the invention having a set of connected objects;

[0010] Figure 2 is a diagram illustrating an embodiment of the invention with two connected objects and illustrating the link therebetween;

[0011] Figure 3 is a diagram illustrating an embodiment of the invention with a plurality of objects and links therebetween;

[0012] Figure 4 is a diagram illustrating an embodiment of the invention having a binary object;

[0013] Figure 5 is a diagram illustrating an embodiment of the invention with a set of objects having a plurality of links;

[0014] Figure 6 is a diagram illustrating an embodiment of the invention with a plurality of binary objects;

[0015] Figure 7 is a diagram illustrating an embodiment of the invention having a single binary object formed from a plurality of sub-objects;

[0016] Figure 8 is a logic diagram illustrating a game according to an embodiment of the present invention; and

[0017] Figure 9 is a logic diagram illustrating a dynamics subsystem according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS AND THE PRESENTLY PREFERRED EMBODIMENTS

[0018] The advantages of using a robust physical dynamics system within a video game are numerous and represent a major improvement over prior art systems. Many different types and genres of video games stand to benefit from improved physical dynamics simulations. For example, a car racing game could be improved by simulating the physical dynamics of the various game objects. Gravity and friction can be used to accurately simulate the behavior of the cars,

leading to a more realistic experience as well as enabling more complex interactions such as cars driving on top of one another. Likewise, a football game could be created with accurate physical interaction between the players. Most current football games use a limited number of pre-recorded animations to depict games events such as tackles. A football game coupled with a sophisticated dynamics system, however, could use the physical dynamics to simulate tackles "on the fly," leading to a limitless number of different possibilities and an enhanced sense of realism.

[0019] The present invention comprises a video game/simulation system with improved physical dynamics interactions among sets of objects. Figure 8 illustrates a high-level view of said game system 100. A game logic subsystem at 106 handles game-specific processing tasks such as implementation of game rules, character movement and artificial intelligence. Input commands from an input device are sent to the game logic subsystem at 112. A graphics subsystem at 104 is operable to generate game specific images such as characters and backgrounds wherein said images are output to a display device at 110. A collision subsystem at 108 handles the processing of collision detection between various game objects. A dynamics subsystem at 102 operatively resolves collision and constraint violations between connected sets of game objects in an improved manner. In order to illustrate the functionality of the dynamics subsystem of the present invention, methods and practices novel to said dynamics system shall first be defined in relation to the attached illustrations.

[0020] Figure 1 shows an example of a connected set of objects. Objects at 1, 3, 5 are connected by links at 7, 9. The term link will henceforth be defined as a set of data representing a connection between two objects. Typically, a link will represent a single point of physical contact between the objects. Therefore, each link includes a position represented by a 3-dimensional vector which defines the point of contact. Each link also has a link orientation defined by three unit-length orthogonal axis vectors (X, Y, and Z). Figure 2 illustrates a link 20 and the associated link orientation X 22, Y 24, and Z 26 axis vectors. Furthermore, each link contains references to both of the connected objects wherein one of the

objects is considered to be the front object and the other the back object. The link orientation Z axis is always coincident to the surface normal (at the point of contact) of the front object. The front object, therefore, defines the contact orientation. In Figure 2, the object at 28 is the front object whereas the object at 30 is the back object. While the link orientation Z vector is defined by the front object surface normal, the corresponding X and Y vectors are arbitrary within the constraint that they must be perpendicular to the Z vector and to one another. Advantages sometimes exist, however, in using more than just the two (X, Y) vectors perpendicular to the normal (Z) vector. The description presented herein details the use of the X and Y direction vectors primarily for the purpose of example and is not intended to limit the scope of the present invention to the use of a set number of orientation vectors.

[0021] The link delta, D, represents the relative speeds of the front and back object at the point of contact. Where V_{front} and V_{back} are 3-dimensional vectors representing the velocities of the front and back objects (at the point of contact) relative to the link orientation, link delta, D, is calculated by:

$$D = V_{front} - V_{back} \tag{1}$$

[0022] A link is considered broken when the z-coordinate of the D vector is greater than zero. A broken link indicates that the objects are about to interpenetrate unless the velocities of the objects are changed.

[0023] A typical connected set of objects is depicted in Figure 3. Each object in a connected set will contain one or more links, denoted by the set $L_1 \ldots L_n$. The object at 40 contains three links: L_1, L_2, and L_3 42. Object links are the primary input/output mechanism for the physics system of the present invention. Many prior art dynamics systems treat objects exclusively as rigid bodies which are capable of rotation and movement. Treating objects as rigid bodies can be advantageous mathematically since the equations of motion are consistent among all objects in a set, a fact which can be exploited to streamline solutions.

Conversely, use of a rigid body system can also limit the types of behaviors and interactions capable of being simulated by such a system. A key aspect of the present invention is that it provides a dynamics system for solving interactions between a set of objects without requiring any explicit foreknowledge of the behavior characteristics of each object. Each object is therefore free to implement its own unique behaviors (i.e., rigid body, soft body, immobile etc.) while being assured of proper interaction with other objects and proper manipulation by the dynamics system of the present invention.

[0024] In order to implement the above mentioned object polymorphism, each object must expose two properties to the dynamics system, velocity and reaction. An object's velocity is defined as an array (set) of three-dimensional velocity vectors V[1] ... V[n] where n is the number of links in the object. Each member of the velocity set, V[x], is defined as the three-dimensional velocity of the object at the position of L_x relative to the link orientation of L_x. The object reaction is defined as the change of object velocity when a specific three-dimensional impulse vector, I, is applied at L_x and is represented by an array of three-dimensional vectors $R^I_x[1] \ldots R^I_x[n]$ where n is the number of links. Each element of the reaction array, $R^I_a[b]$, is defined as the difference in V[b] before and after impulse I is applied to the object at the point of L_a such that:

$$R^I_a[b] = V'[b] - V[b] \qquad (2)$$

where V[b] is the original velocity value and V'[b] is the value of V[b] after impulse I has been applied to the object at the point of L_a. The reaction and velocity arrays allow an object to supply behavioral information in a general way to the dynamics system thereby insuring interoperability between objects of varying behavioral characteristics. As an example, consider a deformable object such as a water balloon. The reaction of the water balloon object (to a specific force) would reflect the change of velocity due to both the deformation and the movement of the object (since both would be affected by the force). Likewise,

consider a second object that is solid. The reaction of the solid object only reflects the movement of the object by the applied force. When the solid object and the water balloon collide, the force of the collision will result in the solid object moving and the water balloon deforming as expected. Likewise, if two water balloons collide, both will be moved and deformed. Furthermore, consider a third object that is a multi-jointed character, such as a person. The reaction of the character object reflects movement of the internal joints as well as the object as a whole. Since the physics engine is doing the same thing in any case (applying force based on the reaction values), it is able to resolve collisions or interactions between any of the object types (character, water balloon, and solid).

[0025] The term binary object, as used herein, refers to a special type of object used by the dynamics system of the present invention. Figure 4 illustrates a typical binary object 50. A binary object always comprises two linked (touching) sub-objects, OBJ1 52 and OBJ2 58. Binary objects have two distinct types of links: external links and cross links. The external links 54, 56, labeled $L_1 \ldots L_n$, represent the points of contact between a sub-object and an external object. The cross links, labeled $C_1 \ldots C_n$, represent the points of contact between the two sub-objects.

[0026] When any of the cross links of a binary object are broken, the binary object may be solved in order to resolve the broken cross links. Solving a binary object involves finding a set of impulse vectors, I[1] ... I[n], which, when applied to the sub-objects at the cross links (i.e., I[1] is applied at C_1, I[2] at C_2, etc.) will ensure that the z component of each cross link delta is less than or equal to zero. The process of determining the correct impulse set essentially involves solving a system of simultaneous linear equations based on the initial delta values and the sub-object reactions. Since, however, simultaneous linear equations are most efficiently solved to zero, the present invention introduces a solution method that allows the controlled solution of linear equation sets to zero and non-zero values while retaining the complexity of solve-to-zero methods.

[0027] A preferred embodiment of the above mentioned solution method for the determination of an impulse set is detailed below. The relevant initial delta

values are stored in array d (d[1] ... d[n] where n is the number of cross links) wherein each element is defined as:

$$d[x] = C_x.D \cdot C_x.Z \tag{3}$$

where $C_x.D$ is the delta value and $C_x.Z$ is the link orientation z-axis vector for cross link C_x. A scalar weight value for each cross link is stored in array w (w[1] ... w[n]) where each weight value is between zero and one. For the sake of example, it should be assumed that the sub-object's links are numbered to correspond to the cross links – i.e., L_1 for OBJ1 ($OBJ1.L_1$) should be equivalent to C_1 and so on. Furthermore, it is herein assumed that the negation of a vector represents a component-wise negation – i.e where K = <K.x, K.y, K.z>, - K = <- K.x, - K.y, -K.z>. A combined reaction value array, B[1...n,1...n], is used to represent the difference in z-axis reaction from the front and back objects at each cross link where:

$$B[x,y] = (FrontOBJ.R^{-Cx.Z}{}_x[y] - BackOBJ.R^{Cx.Z}{}_x[y]) \cdot C_y.Z \tag{4}$$

Arrays A, J and k are further used in the determination of an impulse set and are recursively defined herein as follows:

$$A[p,q,r] = J[p,q] \cdot A[p-1,p,r] + A[p-1,q,r] \quad : p>0,\ q>p \tag{5}$$

$$A[0,p,q] = B[p,q] \tag{6}$$

$$A[p,p,q] = J[p,p] \cdot A[p-1,p,q] \quad : p>0 \tag{7}$$

$$A[0,0,p] = 0 \tag{8}$$

$$J[p,q] = -w[p] \cdot \frac{A[p-1,q,p]}{A[p-1,p,p]} \quad : p>0,\ q>p \tag{9}$$

$$J[p,p] = -w[p] \cdot \frac{d[p] + A[p-1,p-1,p]}{A[p-1,p,p]} \quad : p>0 \tag{10}$$

$$k[p] = J[p,p] + \sum_{q \leftarrow p+1}^{n} (J[q,q] \cdot J[p,q]) \qquad : 1 \leq p < n \qquad (11)$$

$$k[p] = J[p,p] \qquad : p = n \qquad (12)$$

The final values for each member of the above mentioned impulse vector set are then determined by:

$$I[x] = k[x] \cdot C_x.Z \qquad (13)$$

The impulse vector set, I[1] ... I[n], once determined, represents a unique solution for the initial cross link delta set, d[1] ... d[n]. The topography of the solution is controlled by the weight values for each link (w[1] ... w[n]) such that links with a weight value of one are ensured to have a substantially zero z-axis delta value when the solution set is applied, whereas links with weight values less than one may have a non-zero delta value. Furthermore, links having a weight value of zero shall have no impulse applied, i.e., if w[x] = 0, then the length of I[x] must be zero. In a general sense, a link weight, for the purposes of a linear equation solver, independently scales the solution of a link between zero and the value required to completely solve the system of equations. Those in the art may recognize that, with all weight values set to one, the above detailed solution method is functionally equivalent to standard solution methods for linear equation sets that are well known in the art. Indeed, the use of any method for the solution of a binary object is within the scope of the present invention provided that the behavior of said method is properly modified to incorporate link weights as detailed above.

[0028] Binary objects, like all other objects, must be able to provide velocity and reaction arrays to the physics system for all external links. The velocity value at any external link of a binary object is simply equivalent to the velocity value of the corresponding sub-object link. Therefore to provide velocity, the binary object need only provide the velocities of the sub-objects at each external link. The

procedure for providing a reaction array for a binary object is more involved. In order to provide a reaction array for impulse I on external link L_x ($R^I_x[1...n]$), the corresponding reaction array must first be obtained from the appropriate sub-object (the sub-object containing L_x). This reaction array is used to set initial cross link delta values. The binary object is then solved to obtain a reactionary impulse solution array I[1...n] as detailed above. The reactionary impulse set values as well as the initial (L_x) impulse are next applied to each sub-object and the corresponding reactions are summed at each of the external links to produce the combined reaction (velocity change) values which are then returned as $R^I_x[1 \ldots n]$. A binary object, therefore, behaves in such a way as to maintain cross link constraints (such as non-penetration) when external impulses are applied.

In order to provide a substantially linear solution-time, the dynamics system of the present invention recursively divides a connected object set into smaller sets of binary objects until the set is reduced to a single binary object. Figure 5 illustrates an example set of objects. Assuming that the links at 80, 82 between Obj3 and Obj4 are known to be broken, impulses must be applied at these links to resolve the breaks. However, impulses must also be applied at the other links in the set to either resolve their broken status or maintain their non-broken status when Obj3 and Obj4 are resolved. In order to resolve the broken links between Obj3 and Obj4 while ensuring the integrity of the other links, the present invention first reduces the set to a single binary object comprising a recursive set of binary objects wherein Obj3 and Obj4 are contained in the sub-objects. A preferred procedure for the reduction of an object set to a recursive binary object is henceforth presented. Initially, a first iteration is performed where adjacent pairs of objects are combined into binary objects, thereby reducing the number of objects in the set to approximately one half. Figure 6 depicts the first iteration reduction of the example object set in Figure 5 where the set of six original objects is reduced to the three binary objects at 90, 92, and 94. Adjacent pairs of objects are likewise combined over subsequent iterations until the set is reduced to a single binary object. Figure 7 depicts the single binary object, 98, resulting from the application of the above mentioned procedure. It should be recognized by

those in the art that the connected object set may be reduced to a target binary object in an alternate order and by alternate procedures than those depicted herein. The scope of the present invention does not limit the conversion of a connected object to a target binary object to a specific order nor is the topology of each binary object likewise limited.

[0029] Once the target binary object is found, it must be solved to determine the impulse set required to fix the broken cross links. First each sub-object, starting with the most deeply nested binary objects and proceeding outward, is recursively solved to ensure that sub-object links are non-broken. The target binary object is next solved to determine the impulse set required to fix the cross link breaks. The cross link impulses are then applied to the sub-objects. Since the sub-objects are themselves binary objects, impulses applied at the external links will in turn cause reactionary impulses to be applied at each of the sub-objects. In this manner, the proper impulses are applied at all of the links in the object set to maintain their non-broken status. Since all reactions will only be calculated in the direction of the link z-axis and since the reaction values can be considered proportional to the applied force (i.e., $R^{x \cdot I}{}_a[b] = x \cdot R^I{}_a[b]$), reactions need only be calculated once for each link and can be subsequently stored in memory for later use. Because the binary object solution time depends on the maximum number of cross links between any two objects (which can be considered a constant), the number of solution steps is thus linearly dependent on the total number of objects in the object set. The above detailed method of recursively dividing and solving the object set therefore has the advantages of providing a substantially linear solution time as well as being able to provide a solution for any object set topology (i.e., closed loops). More importantly, since solutions and reactions for binary objects at the same depth can be computed in parallel, said recursive method provides the further advantage of a parallelizable solution to an arbitrary object set. Since any connected object set can be reduced to a binary object with a depth at or near the base-2 log of the set size (lg n), a parallel implementation of the above mentioned recursive solution method is able to solve an object set with an O(lg n) time complexity, thereby presenting a dramatic improvement over prior

art approaches. Prior art approaches employing LCP/pivoting solutions solve the set in discreet steps rather than at once and therefore cannot be effectively parallelized. Furthermore, recursively based prior art approaches cannot solve for arbitrary set topologies and are likewise unable to maintain contact and/or friction constraints.

[0030] To demonstrate an example implementation of the parallel solution mentioned above, consider a system with four processors where each processor is capable of solving a set of linear equations. A recursive binary object is next considered which comprises two binary sub-objects, each of which further comprises two binary sub-objects for a total of eight objects. Each of the four processors then solves each of the four depth-3 binary objects in parallel – calculating both the initial solution and the reactions. This operation can be considered to take a constant amount of time (since the number of cross links in each binary object is substantially constant). Once the depth-3 solutions are calculated, two of the processors are then used to solve each of the two depth-2 binary objects in parallel – likewise taking a constant solution time. Finally, one of the processors solves the single depth-1 binary object (also in constant time), yielding the solution to the system. Thereby a system of eight connected objects is solved within the time frame of three cross link solutions, providing a very fast and efficient solution.

[0031] A second aspect of the dynamics system of the present invention is the ability to enforce constraints such as hard contact and friction. Hard contact constraints disallow object interpenetration while ensuring that adhesive force is not used to hold objects together (i.e., only positive impulse is allowed at any link). Some prior art approaches attempt to minimize systems of constraint equations to enforce contact constraints whereas other systems solve the contact constraints as a linear complementary problem (LCP). Either approach can lead to a high order solution complexity in some cases. In order to ensure a parallelizable, substantially linear time solution for implementing hard contact constraints, the dynamics system of the present invention calculates a set of link weights which produce a solution adherent, within a tolerance, to the contact constraints. Since

the correct set of link weights cannot be known a priori for a particular configuration, the dynamics system of the present invention iteratively adjusts the set of link weights over multiple solutions. A preferred method of the present invention for implementing contact constraints for a given object set is detailed below. Initially, all link weights are set to 0.5. The object set is then repeatedly solved over two or more iterations. At each iteration, the link weights are increased by a set amount for all links with a total impulse greater than zero. Likewise, link weights are decreased for all links with a total impulse less than zero. After each iteration, constraint violation error values (such as maximum interpenetration velocity and maximum negative impulse) are calculated. If the constraint violation errors are acceptably small (i.e., below a set tolerance), the process is terminated and the impulse and weight values from the last solution are used. The process can likewise be terminated after a set number of iterations. In addition, per-iteration weight adjustment can be directionally skewed to favor the maintenance of one constraint over the other. For example, weight values can be set to only increase after a set iteration is reached – thereby ensuring the non-penetration constraint is completely satisfied at the expense of the non-negative impulse constraint. The practice of iteratively adjusting link weights gives the dynamics system of the present invention several advantages over prior art approaches. Chiefly, said practice enables the estimation of a dynamics solution, within a selectable level of accuracy, while retaining the ability to completely enforce certain constraints (such as non-penetration). Furthermore, said practice enables the ability of a dynamics solver to implement constraints which vary continuously over a set of inputs (such as velocity or force), consequently resulting in a dramatic increase in the types of behaviors that can be simulated.

[0032] Friction constraints are generally more difficult to enforce in dynamics systems. In order to accurately simulate physical interaction, constraints for both static and dynamic friction must be enforced. Both types of friction involve a frictional force/impulse applied at the lateral plane of contact (the X/Y plane of the link orientation) and are subject to the constraints that the relative lateral motion between the objects is zero or that the frictional force is equal to the total normal

force scaled by a friction coefficient. In many prior art approaches, dynamic friction constraints are solved in a manner distinct from static friction and can consequently lead to unsolvable systems. The present invention provides a system to handle static and dynamic friction constraints in the same manner whereby unsolvable systems are not introduced.

[0033] In order to implement frictional constraints in the present invention, the above detailed binary object solution method must be amended to solve each link along all three axes rather than only the z-axis. In order to solve on all three axes, function t(x) is first defined where:

$$t(x) = (x + 2) / 3 \quad : x \bmod 3 = 1 \tag{14}$$

$$t(x) = (x + 1) / 3 \quad : x \bmod 3 = 2 \tag{15}$$

$$t(x) = x / 3 \quad : x \bmod 3 = 0 \tag{16}$$

Vector array P[1...3n] is further included wherein:

$$P[x] = C_{t(x)}.X \quad : x \bmod 3 = 1 \tag{17}$$

$$P[x] = C_{t(x)}.Y \quad : x \bmod 3 = 2 \tag{18}$$

$$P[x] = C_{t(x)}.Z \quad : x \bmod 3 = 0 \tag{19}$$

The initial delta array d is redefined as d[1...3n] wherein each element is defined as:

$$d[x] = C_{t(x)}.D \cdot P[x] \tag{20}$$

Likewise the combined reaction value array B is redefined as follows:

$$B[x,y] = (\text{FrontOBJ}.R^{-P[x]}{}_{t(x)}[t(y)] - \text{BackOBJ}.R^{P[x]}{}_{t(x)}[t(y)]) \cdot P[y] \tag{21}$$

In addition, each link is expanded to include three scalar weight values (Wx, Wy, Wz) corresponding to each of the three link axes. The w array is consequently redefined as:

$$w[a] = C_{t(a)}.Wx \quad : a \bmod 3 = 1 \tag{22}$$

$$w[a] = C_{t(a)}.Wy \quad : a \bmod 3 = 2 \tag{23}$$

$$w[a] = C_{t(a)}.Wz \quad : a \bmod 3 = 0 \tag{24}$$

The A and J arrays remain as previously defined whereas the k array is henceforth redefined as:

$$k[p] = J[p,p] + \sum_{q \leftarrow p+1}^{3n} (J[q,q] \cdot J[p,q]) \quad : 1 \leq p < 3n \tag{25}$$

$$k[p] = J[p,p] \quad : p = 3n \tag{26}$$

The final impulse vectors, I[1...n], are lastly determined by:

$$I[x] = (k[x] \cdot P[x]) + (k[x+1] \cdot P[x+1]) + (k[x+2] \cdot P[x+2]) \tag{27}$$

The above detailed modified binary object solution method is functionally the same as the previously defined solution method with the exception that variables are given for each of the three axes (X, Y, and Z) rather than only the z-axis. Likewise, the distinguishing point of the above detailed method is the use of one or more substantially continuous link weights for each link. Therefore, the above detailed preferred solution method may thus be altered or superseded within the scope of the present invention provided that link weights are handled in a functionally similar manner.

[0034] A preferred method of the present invention for implementing contact and friction constraints for a given object set is henceforth detailed. As with contact constraints, link weight values are iteratively adjusted over multiple solutions. Initially, all link weights (Wx, Wy, and Wz) are set to 0.5. After each

iteration, link weights are adjusted based on the impulse applied. Link weight z values (Wz) are adjusted based on the z-axis (normal) impulse as previously detailed wherein positive impulse increases the weight while negative impulse decreases it. Lateral link weights (Wx, Wy) are adjusted to produce the proper lateral impulse in relation to the normal impulse and coefficient of friction. With N being the total normal impulse, F being the total lateral impulse, and c being the coefficient of friction for a given link, the adjusted x-axis link weight value (Wx') is calculated from the previous x-axis value (Wx) and z-axis value (Wz) by:

$$Wx' = \frac{c \cdot N \cdot Wx}{F \cdot Wz} \tag{28}$$

[0035] The Wx' value is then clamped to a zero-to-one range. Alternate embodiments additionally limit the maximum change in link weight for each iteration. The Wy' value is then set equal to Wx'. After each iteration, friction constraint error values (such as the largest difference between applied friction impulse and optimal impulse) are calculated along with the aforementioned contact constraint error values. The process may then terminate when all error values are below a preset tolerance or when the number of iterations reaches a preset maximum. Once the process is terminated, impulse and weight values from the current iteration are used as the final results. Since static and dynamic friction are both enforced by the lateral link weights and since link weights, at most, reduce the relative velocity to zero, the maintenance of dynamic and static friction constraints by the use of link weights, therefore, cannot introduce insolvability into the system. Alternate embodiments, as will be recognized by those in the art, use other methods to calculate adjusted link weight values at each iteration including, but not limited to, quadratic and cubic estimation strategies.

[0036] A further aspect of the present invention in relation to the previously detailed iterative approach to constraint maintenance is the ability of any object in the system to maintain internal constraints wherein said constraints may vary continuously with the impulse applied to the object. Specifically, during each

solution iteration, an object may alter its internal behavior, and thus its reaction(s), dependent on the total impulses applied. This is, in essence, what binary objects do through the adjustment of link weights. The mechanism of altering behavior at each iteration, however, is not limited to binary objects. Within the present invention, other objects may implement said mechanism to simulate complex behaviors which has been heretofore difficult or impossible to implement in prior art dynamics systems. As an example, consider a large structure such as a bridge represented by a single object comprised of a number of connected pieces. With nominal force applied, the pieces of the bridge remain firmly connected and the bridge is immobile, thereby producing a zero reaction (since the forces will not change the velocity of any part of the bridge). If a larger force is applied, and the tension between some of the connected pieces becomes greater than a pre-set maximum, the bridge can break at some of the connections producing an alternate reaction with the broken pieces moving from the applied force. In a likewise manner, collisions between multiple breakable objects can be simulated, with each object breaking realistically under the collision forces, thereby providing a more realistic and immersive gaming experience to the end user.

[0037] While the detailed description above teaches the applicability of a weight-based iterative solution method to the resolution of contact and frictional constraints over points of hard contact between objects, it should be noted that said solution method is likewise applicable in solving alternate types of constrained links between objects. For example, links wherein all link weight values are set to one are functionally equivalent to rotation joints with three degrees of freedom. Likewise a link with a weight of one along the normal direction is able to implement a point on surface contact constraint. Breakable constraints can also be solved by the above mentioned iterative solution method where, for example, link weights are initially set to one but are reduced on subsequent iterations if the net impulse on the link is greater than a predetermined limit value. A further beneficial aspect of the present invention is the ability of individual objects to implement alternate constraints by any method while interactions with other objects are able to be handled by the above mentioned solution method. For

example, an object may comprise a multi-jointed body (such as a humanoid figure) wherein algorithms local to the object control the behavior and resolution of the joints. Provided that the object's internal behavior (when impulses are applied) is consistent with the behavior implied through its reaction(s), said object is able to interact with other objects (such as rigid bodies) through the above detailed solution methods. Said methods are likewise applicable to handle internal collisions within an object without inherent modification of the object behavior (i.e., the same reaction algorithm can be used for both internal and external contact links).

[0038] Figure 9 depicts a logical view of the dynamics subsystem 120 of the game system of the present invention. The purpose of the dynamics subsystem is, given one or more connected object sets, to produce a set of link impulses such that application of said impulse sets shall substantially maintain one or more constraints, such as non-penetration and/or friction, between objects in said connected object set(s). At 132, object set information is provided to a binary division subsystem 122 operable to convert one or more connected object sets into recursive binary objects as previously detailed. In some embodiments, multiple processors are utilized by the binary division subsystem to reduce the conversion time required by large connected object sets. Recursive binary object data is then passed at 124 to a solver subsystem 126 wherein said subsystem is operable to iteratively compute solution impulse sets for one or more binary objects. In a preferred embodiment, the solver subsystem comprises multiple processors to simultaneously compute solution and reaction impulse sets for multiple binary objects. The solver subsystem iteratively solves the given binary object(s) wherein link weight values are adjusted at each iteration as previously defined. In order to incorporate object polymorphism and iterative object behavior modification, object behavior data is transferred to and from the solver subsystem at 134. Impulse set and object reaction information is transferred at 128 to an impulse combination subsystem 130. The impulse combination subsystem is operable to combine external and cross link impulse values for one or more binary objects to produce a set of total impulse values for substantially each link in a

connected object set. Some embodiments employ multiple processors to decrease the time taken to calculate said total impulse value set. The set of total impulse values is then output from the dynamics subsystem at 136.

[0039] While the dynamics system of the present invention has been detailed herein using the examples of impulse and velocity, it should be noted by those in the art that the use of said system for the solution of object sets where impulse is replaced by force and velocity by acceleration is functionally equivalent with a few distinctions. Firstly, while the aforementioned cross-link delta values (d[1...n]) designate differences in acceleration rather than velocity, exceptions are made in the case of dynamic friction. For links with a non-zero relative lateral velocity (i.e., links subject to dynamic friction), said lateral velocity should be scaled and combined with the lateral acceleration in the cross-link delta value(s) to represent the effects of the acceleration over a set time period (such as frame length). Likewise in links with a negative relative normal velocity (where the linked objects are moving apart), a scaled normal velocity may be combined with the relative normal acceleration value in the cross-link delta. Using the aforementioned combined cross link normal delta will ensure no force is applied at links with a negative velocity past a set limit while force is still applied at links with negligible relative velocities, thereby avoiding redundant solutions.

[0040] The detailed description presented herein teaches a system and method whereupon physical interactions are simulated among an arbitrary set of objects in a low-order time frame with respect to the total number of objects in the set and wherein hard contact and friction constraints are maintained. It should be noted by those in the art that methods and operations detailed above are presented herein for the purpose of example and may thus be modified and/or optimized to an extent without departing from the scope of the present invention as defined by the appended claims and equivalents. It should also be noted that the game system of the present invention and the various subsystems presented therein may be implemented as software instructions running on a general purpose CPU and/or by custom computer hardware dedicated to performing one or more of the above-identified operations without departing from the scope of the invention.

[0041] It is therefore intended that the foregoing detailed description be regarded as illustrative rather than limiting, and that it be understood that it is the following claims, including all equivalents, that are intended to define the spirit and scope of this invention.